НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль

(підпис) (ініціали, прізвище)

“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Комп’ютерні науки“

на тему Автоматизована система класифікації звуковипромінюючих об’єктів за їх спектральнимихарактеристиками

Виконав: студент __4__ курсу, групи ТР-61

Буренок Артем Ігорович _____

(прізвище, ім’я, по батькові) (підпис)

Керівник ___доцент к.т.н. Варава І.А._____ _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____доцент к.т.н. Корольов А. П._____ _____

(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____

(підпис)

Київ – 2020

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

зі спеціальності 122 Комп’ютерні науки та інформаційні технології
за спеціалізацією Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)

”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту
Буренку Артему Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи: Автоматизована система класифікації звуковипромінюючих об’єктів за їх спектральними характеристиками

керівник роботи__ доц. Варава І. А. к.т.н._
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020 р. № 1268-с

2. Строк подання студентом роботи 15 червня 2020 р.

3. Вихідні дані до роботи : мова програмування Python, бібліотека Keras, середовище розробки Jupyter Notebook.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Проаналізувати існуючі сучасні алгоритми для класифікації звуку. Розробити

алгоритм на базі багатошарових нейронних мереж для розв’язання задачі класифікації звуку. Дослідити отриманий алгоритм. Розробити додаток для подачі результатів користувачеві.

5. Перелік ілюстративного матеріалу:

Робота: презентація 12 слайдів

Робота: Матриця заплутувань

Робота: Jupiter Notebook частини проекту

Результати роботи: інтерфейс

Результати роботи: інтерфейс додатку

7. Дата видачі завдання ”10” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
.	Затвердження теми роботи	10.10.2019	
.	Вивчення та аналіз задачі	10.10.2019-20.01.2020	
.	Розробка архітектури та загальної структури системи	20.01.2020-20.02.2020	
.	Розробка структур окремих підсистем	20.02.2020-04.03.2020	
.	Програмна реалізація системи	04.03.2020-26.04.2020	
.	Оформлення пояснювальної записки	26.04.2020-06.06.2020	
	Захист програмного продукту	04.06.2020	

.			
.	Передзахист	10.06.2020	
.	Захист	17.06.2020	

Студент _____ _Буренок А. І. ___
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Варава І. А. _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Актуальність теми

Із урахуванням стрімкого розвитку мікроконтролерів, появи дешевої елементної бази для датчиків та систем автоматичного генерування сигналів здатність формування датасетів гідроакустичних сигналів зростає. Як результат, зростає необхідність в розвитку систем, які націлені на обробку такої інформації автоматизовано. Важливою задачею є задача класифікації таких звукових сигналів, це важливо для сенсорних систем, які обробляють інформацію, що поступає з датчиків в реальному часі та не мають спостережувача-експерта, який міг би класифікувати сигнали сам. На сьогоднішній день існує багато алгоритмів, які є направленеми на вирішення такої задачі, але це дослідження всерівно залишається актуальним.

Об’єктом дослідження є програмні засоби класифікації звукових гідроакустичних сигналів з використанням нейронних мереж для створення відповідних моделей.

Предметом дослідження є алгоритми призначені для розпізнавання звуку за допомогою машинного навчання, сконцентровуючись особливо на нейронних мережах, методи знаходження характеристик звуку, які найкраще підходять для алгоритмів в даній предметній області.

Мета дослідження:

Метою є підвищення ефективності розв'язання задачі класифікації гідроакустичних сигналів за допомогою багатошарових нейронних мереж для задачі класифікації.

Задачі дослідження:

1. Аналіз сучасних алгоритмів для класифікації звуку.

2. Розробка алгоритму на базі багатошарових нейронних мереж для

розв’язання задачі класифікації звуку.

3. Дослідження отриманого алгоритму.

Ключові слова. Штучні нейронні мережі, мел-кепстральні коефіцієнти, аудіосигнали

ANOTATION

Actuality of theme

The rapid development of microcontrollers, the emergence of a cheap element base for sensors and automatic signal generation systems, the ability to form datasets of sonar signals is growing. As a result, there is a growing need to develop systems that aim to process such information automatically. An important task is the task of classifying such audio signals, this is important for sensor systems that process information coming from sensors in real time and do not have an expert observer who could classify the signals itself. Nowadays there are many algorithms that are aimed at solving this problem, but this study is still relevant.

The object of the research is software tools for classification of sound sonar signals using neural networks to create appropriate models.

The subject of the study are algorithms designed for sound recognition using machine learning, focusing especially on neural networks, methods for finding sound characteristics that are best suited for algorithms in this subject area.

The aim of the study:

The aim is to increase the efficiency of solving the problem of classification of sonar signals using multilayer neural networks for the problem of classification.

Research objectives:

1. Analysis of modern algorithms for sound classification.

2. Development of an algorithm based on multilayer neural networks to solve the problem of sound classification.

3. Research of the received algorithm.

Key words: artificial neural networks, chalk-cepstral coefficients, audio signals.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MFCC (англ. Mel-frequency cepstral coeficients) — коефіцієнти за допомогою яких можна представити енергію спектра сигналу, вона базується на сприйнятті звуку людських вухом;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

DAT (англ. Digital Audio Tape) — формат файлу цифрової аудіо доріжки;

ЗМІСТ

ВСТУП

Класифікація звукових даних - це зростаюча область досліджень з численними прикладами реальних додатків у реальному світі. Незважаючи на те, що існує велика кількість досліджень у суміжних аудіо-сферах, таких як розпізнання мови чи музики, робота над класифікацією сигналів в гідроакустиці порівняно не велика. Так само, спостерігаючи за останніми досягненнями в галузі класифікації зображень, де нейромережі використовуються для класифікації зображень з високою точністю та масштабом, виникає питання про застосованість цих методів в інших сферах, таких як класифікація звуку, де інколи трапляються дискретні звуки.

Існує декілька підходів до класифікування звукових сигналів найпопулярнішими із них є алгоритми на основі методів глибокого навчання та за допомогою інструментів DSP.

Мета цього проекту - застосувати методи глибокого навчання до класифікації звуків навколишнього середовища, зосередившись саме на гідроакустичних сигналах.

Вхідними даними до програмного продукту є гідроакустичні сигнали, вихідними дані про належність сигналу до певного класу з точністю, яка залежить від навчаної моделі.

1 ПІДХОДИ ДО КЛАСИФІКАЦІЇ ЗВУКУ

1.1 Характеристики сигналів, які використовуються в цілях класифікації

1.1.1 Загальний огляд характеристик звуку

Існує велика кількість різних характеристик звуку, які використовуються для роботи з акустичними даними [1].

До списку задач, які використовують ці характеристики для класифікації, входять задачі розпізнавання мови, задачі створення схожих сигналів, створення заголовків для аудіо різного формату,та інші.

1. Спектрально-часові ознаки:

1.1 Спектральні ознаки:

- середнє по спектру заданого звукового сигналу;

- середні значення спектру в нормалізованому вигляді;

- обрахований час сигналу під час перебування в смугах спектра;

- нормалізоване значення часу перебування в смугах;

- медіана спектра аудіо в смугах;

- обраховане значення потужності спектра в смугах;

- інші ознаки, які були отримані при маніпуляціях зі спектром.

1.2 Часові ознаки:

- довжина сегменту, тривалість звуку;

- висота заданого сегменту.

Спектрально-часові ознаки описують фізико-математичну суть сигналу, беручи до уваги наступні характеристики:

- Періодичних ділянок хвилі сигналу;

- Неперіодичних ділянок хвилі сигналу(шумових); ділянок, які не містять в собі пауз

2. Кепстральні ознаки сигналу:

-мел-частотні кепстральні коефіцієнти(MFCC);

-коефіцієнти лінійного передбачення, які скоректовані через факт нерівномірності чутливості вуха людини;

-коефіцієнти потужності частоти регістрації;

-коефіцієнти лінійного передбачення спектру;

-коефіцієнти лінійного передбачення кепстра.

3. Амплітудно-частотні ознаки:

-інтенсивність та амплітуда;

-енергія сигналу;

-частота головного тону (чот);

-формантні частоти сигналу;

-джиттер - модуляція частоти основного тону;

-шиммер – модуляція амплітуди основного тону;

-базисна радіальна ядерна функція.

За допомогою амплітудно-частотних значень можна отримати ознаки , які змінюються в залежності від параметрів перетворення Фур'є або малих зміщень вікна при вибірці.

Якщо говорити словами науки акустики, то сигнал це поширювання в повітрі складних за структурою звукових коливань, вони описуються частотою (число коливань за одну секунду), інтенсивністю (амплітуда коливання) і довжиною. Амплітудно-частотні ознаки описують інформацію для розпізнавання звукового сигналу.

4. Ознаки нелінійної динаміки:

-відображення Пуанкаре;

-рекурентний графік;

-характеристичне значення Ляпунова;

-фазовий портрет;

Для ознак нелінійної динаміки сигнал розглядають як скалярну величину, її можна спостерігати в голосовому тракті будь-якої людини.

Сам процес утворення мови вважають нелінійним і аналізують методами нелінійної динаміки[1]. Завданням нелінійної динаміки є знаходження і дослідження моделей до яких входять математичні моделі і системи реального світу, вони виходять з найтиповіших ідей про властивості елементів, з яких складається система, а також законів по яким вони взаємодіють.

На даний момент всі методи нелінійної динаміки будуються на фундаментальній математичній теорії, в її основі лежить відома теорема Такенса, вона підводить математичну основу під деякі ідеї нелінійної авто регресії, а також доводить можливості відтворення фазового портрета атрактора використовуючи часовий ряд або використовуючи всього одну його координату. (Атрактор це множина точок або внутрішній простір у фазовому просторі, до нього повинна наближаєтися фазова траєкторія після того як перехідні процеси загаснуть.)

Оцінки характеристик сигналу з отриманих траєкторій використовують в будуванні детермінованих нелінійних фазово-просторових моделей часового ряду. Для виявлення відмінності у формі атракторів можна користатися для правилами і

ознаками для діагностики, вони дозволяють, для прикладу, розпізнавати і точно ідентифікувати великого спектру емоції в сигналі наповненому емоційним станом.

Найчастіше при класифікаціях використовуються спектральні ознаки та мел-кепстральні характеристики (MFCC).

MFCC – це коефіцієнти за допомогою яких можна представити енергію спектра сигналу, вона базується на сприйнятті звуку нашим вухом.

Доведено, що чутливість людини до звукового сигналу залежить від його частоти: чим нижче частота, тим чутливість вище. У 1937 році була виведена формула, за допомогою якої можна перевести частоту в герцах в частоту в Гаммелах[2].

Мел - це одиниця виміру частоти певного звуку, створена після статистичної обробки великого числа даних про сприйняття висоти тонів сигналів.

Формули для переведення частоти у герцах (f) і у Гаммелах (m) можна переглянути знизу:

$$m = 1127.01048 \ln(1 + f/700)$$

$$f = 700(e^{m/1127.01048})$$

Сигнал може бути поданий згорткою двох функцій: вихідного сигналу, а також фільтра, його параметри ми і хочемо оцінити. Потрібно виділити ці компоненти. Щоб цього досягнути потрібно перетворення, яке могло би перетворити згортку в суму:

$$x * h \rightarrow x + h,$$

Для цього вводиться перетворення кепстра[2].

$$C[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln\left|X(e^{iw})e^{iwn}\right| dw \text{ - дійсний вектор}$$

$$C[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln(X(e^{iw}))e^{iwn} dw \text{ - комплексний вектор}$$

Схему перетворення сигналу можна записати в наступному вигляді:

Перетворення Фур'є: $x * h \rightarrow XH$

Перетворення кепстра: $XH \rightarrow X' + H'$

Зворотне перетворення Фур'є: $X' + H' \rightarrow x' + h''$

Тож розділення сигналу на джерело та фільтр відбувається автоматично. Використання MFCC має наступні переваги:

- Береться до уваги хвильовий характер звукової хвилі(за рахунок користування спектровою інформацією в неявному вигляді);

- через проектування заданого спектру на мел-шкалу можна виділити найважливіші частоти для сприйняття людським вухом;

- кількість мел-кепстральних коефіцієнтів аудіо можна варіювати. За рахунок цього можна масштабувати обсяг інформації, яка оброблюється за рахунок “стиснення” фрейму.

1.2 Популярні алгоритми для класифікації звуку

Класифікація належить до одного із розділів машинного навчання, він присвячений вирішенню наступної задачі: на вході є множина об'єктів, вони розділені певним чином на деяку кількість класів[3]. Обмежена множина з заданими назвами класів називається початковою вибіркою. Належність інших об'єктів до заданих класів не є відомою. Потрібно побудувати алгоритм, який є здатним для класифікування будь-якого об'єкту із початкової множини.

Класифікувати об'єкт — означає поставити його у співвідношення до відповідного класу, встановити відповідність між заданим об’єктом та класом із множини допустимих класів.

В математичній статистиці такі задачі інколи також називають задачами дискримінантного аналізу.

В розділі машинного навчання задачі класифікації відносять до розділу навчання із вчителем. Може існувати навчання і без вчителя, тоді розподіл всіх об'єктів початкової вибірки на класи не вказаний, та необхідно прокласифікувати об'єкти спираючись тільки на схожості їх одним з одним. В такому разі прийнято говорити вже про задачу кластеризації її ще називають задачею таксономії, а класи відповідно називають кластерами або таксонами. Ці задачі часто є схожими.

Класифікацію розділяють в залежності від особливостей вхідних даних тобто класів, які оброблюються в процесі.

По вхідним даним класифікація розділяється на наступні класи:

1. По ознакам об’єкту – найчастіше використовується на практиці. У кожного об’єкту є список своїх характеристик, їх ще називають ознаками. Ознаки можна поділити на числові та нечислові.

2. За відстанню між заданими об'єктами. Кожен із об'єктів може описуватися відстанями до інших об'єктів взятих із початкового датасету. Із подібним типом даних на вхід працює досить мало методів, до них входять метод найближчих сусідів, а також метод парензівного вікна і метод потенційних функцій.

3. Часовий ряд чи сигнал це послідовність вимірів по часу. Кожен раз вимірювання можна представити вектором, цифрою, або у загальному випадку можна представити описом об'єкта який досліджується в момент дослідження.

4. Зображення або відеоряд.

Існують і інші складніші випадки, вхідні дані можуть представляти з себе більш складну структуру даних такі як графи, такслі та інші. Зазвичай, ці випадки можна звести до першого або другого за допомогою різноманітних методів до обробки даних та виявлення їх ознак.

За класами задачу класифікації визначають як:

1. Класифікація із двома заданими класами. Це найбільш простий для

реалізації вид класифікації, він є основою для розв’язування уже більш складних видів класифікації.

2. Класифікація із заданими більше ніж двома класами. Коли в задачі постає потреба в класифікації великої кількості об’єктів до великої кількості класів(наприклад задача розпізнавання конкретного типу рослини, яких тисячі видів), тоді задача класифікації набуває складнішого вигляду.

Поділити класифікацію на види інакшим способом можна на:

1. Класифікація з наявними класами, що ніколи не перетинаються між собою.

2. Класифікація з наявними класами, що можуть перетинатися. Тобто об'єкт може бути одночасно класифікований до декількох типів.

3. Класифікація з наявними класами без чіткої назви. Суть задачі зводиться до визначення відсотку належності до кожного класу, зазвичай фінальний результат представляється у вигляді дійсного числа від 0 до 1.

Задача класифікації звуку це класифікація по ознакам об’єкту (можна сказати, що класифікацію зображень теж коректніше віднести до класифікації по ознакам, тому що значення пікселя можна розглядати як ознаку), багатокласова (частіше всього) та з класами, які не перетинаються(коректніше було б працювати з задачею класів, які перетинаються, але вона є значно складнішою, і що унеможливлює задачу вимагає відповідних вхідних даних).

Для класифікації звукових сигналів можна скористатися будь-яким алгоритмом, який використовується для розв’язання задачі класифікації.

Розглянемо деякі із алгоритмів класифікації:

1.2.1 Метод опорних векторів.

Метод опорних векторів (англ. support vector machine, SVM), це машинний алгоритмом, який навчається на заданих прикладах і використовується в задачі класифікації об’єктів[4].

Освоною SVM є математична модель, яка представляє із себе алгоритм максимізації якоїсь математичної функції відносно певного набору даних.

Для правильного розуміння, як працює SVM, потрібно знати про поняття:

– розділяюча гіперплощина;

– гіперплощина максимальної межі

– м’яка межа;

– функція ядра.

Розділяюча гіперплощина це математична сутність, що відділяє класи з однаковими ознаками.

Можна екстраполювати процедуру до багатовимірних просторів, розмірності яких набагато вищі від третьої. В загальному лінія, якою відділяються елементи класів називається багатовимірна гіперплощина.

SVM має різницю з іншими методами класифікації в тому, що за допомогою нього можна вибрати оптимальне розташування гіперплощини. Гіперплощину обирають так, щоб вона була розташованою на максимально можливій відстані від елементів кожного з класів, тобто в середині обмеженої зони, яка відділяє між собою елементи класів.

Це представляє собою інше ключове поняття, а саме гіперплощина максимальної межі.

Об’єкти, які беруть участь у класифікації не завжди можна розділити гіперплощиною. В реальних практичних системах завжди будуть мати місце похибки у даних, через це гіперплощина не зможе виконати поділ точно. Саме тому в методі SVM є допустима похибка класифікації, яку називають м’якою межею.

Метод опорних векторів не повинен робити помилки класифікації спостережень. Саме тому нам необхідний ще один параметр який буде вказувати на кількість об’єктів, які можуть бути класифіковані неправильно, а також відстань на якій вони можуть бути розташовані від неї. Інакше кажучи, ми вводимо м’яку межу похибки з двох боків площини, яка розділяє класи. Це зв’язано з тим, що об’єкти, які потрібно класифікувати, можуть бути не розділені тільки лінійно. Трапляється таке,

що ці об’єкти не допускають можливості на таке розділення. Для вирішення цієї проблеми використовуються функції ядра, які переносять вхідні дані з маловимірного простору у більш багатовимірний. При вірному виборі функій ядра ми зможемо розділити об’єкти лінійно у багатовимірному просторі. Отже функції ядра виконують роль простору, який спрямовує.

1.2.2 Метод логістичної регресії

Логістична регресія (англ. Logit model) – це така статистична модель, яку використовують щоб підрахувати ймовірність події через підгонку вхідних даних до логістичної кривої. Ще її можна визначити як регресійну модель, яка може передбачати категоріальну змінну[3]. Попри своїй назві, цей вид регресії використовують для класифікаційних задач, а не регресійних.

Даний вид регресії застосовують також для обрахування ймовірності деякої події по значенням багатьох ознак. В цьому випадку використовується залежна змінна y, вона приймає на вхід одне з двох даних значень - зазвичай, це числа 0 (якщо не відбулося події) і 1 (якщо відбулася), і великої кількості незалежних ні від чого змінних — реальних значень x_1 , x_2 , x_3 , x_4,. .. , x_n , на їх основі потрібно обрахувати імовірність прийняття незалежною змінною значень.

Спочатку робиться припущення, що імовірність того що настане подія y = 1 дорівнює:

$$P\{y=1|\,x\}=f(z) \tag{1}$$

, де $z=\theta^{\mathrm{T}}x=\theta_0+\theta_1 x_1+...+\theta_n x_n$ x та θ - вектори незалежних змінних $1,x_1,x_2,...,x_n$ і параметрів регресії - $\theta_1,\theta_2,...,\theta_n$ відповідно, а $f(z)$ це логістична функція(сигмоїд)

$$\mathrm{f(z)}=\frac{1}{1+e^{-z}} \tag{2}$$

1.2.3 Метод k-найближчих сусідів(k-means)

Метод k-найближчих сусідів[3]. Являється метричним методом автоматизації. При використанні даного методу для класифікації всім об’єктам відноситься клас, який є найчастіше зустрічається серед його k сусідів, класи цих сусідів були визначені раніше. Метод можна модифікувати, використовуючи більшу вагу до класів які розташовані ближче та меншу для тих які далі.

Якщо використовувати метод для регресії, то об'єкту спочатку присвоюють середнє по k об'єктам, які знаходяться найближче до нього, та у яких значення належності до класу вже відомі .

Є також імовірнісний варіант даного методу. Оцінювати ймовірність того, що ознака об’єкту належить саме класу P , можна як частку k найближчих класів сусідніх об’єктів у класі P .

Сам параметр k у цьому методі вибирається по знанням про задачу класифікації спираючись на досвід розв’язування схожих задач. Частіше всього, параметр k роблять непарним, це зменшує імовірність віднести об’єкт одразу до двох класів.

Даний алгоритм підходить для вибірок з багатьма атрибутами. Щоб це зробити треба ввести функцію дистанції. Дистанція може бути вибрана будь яким чином дистанція в евклідовому просторі, манхетенська відстань чи інші її види.

Основним недоліком цього алгоритму є те що необхідно одночасно берегти у пам'яті велику кількість даних. І хоча в сучасному світі проблема недостатності пам'яті відходить на задні місця у деяких випадках це може стати голосом проти використання цього методу. Найголовнішою перевагою даного методу звісно є швидкість його виконання, якщо порівнювати його з іншими методами, наприклад з штучними нейронними мережами, деревом рішень чи машиною опорних векторів.

1.2.4 Метод з використанням GMM класифікатор

Модель Гаусової суміші являється ймовірнісною моделлю, вона опирається на припущення, про те що спостереження, які відносяться до конкретних класів, створюються із суміші дискретного значення гаусових розподілів із зарані невідомими параметрами [3]. Цю модель можна розглянути, як узагальнення до k-means алгоритму із включенням додаткової інформації про структуру вхідних даних, а також центри їх розподілів.

Цікавість GMM моделі полягає в тому, що дана модель бере до уваги вхідні дані як результат комбінації лінійно заданої кількості гаусових розподілів, коли інші методи, так само як і статичні використовують один певний розподіл, або ж взагалі просто розглядають взаємовідносини між класами.

З урахуванням того, що саме нормальний розподіл є дуже розповсюдженим на практиці в реальному світі, увага саме до цього методу є логічно великою. Крім цього, із центральної граничної теореми витікає, що у випадку, якщо робота виконується із складними даними, а також кількість невідомих в датасеті факторів велика, тоді використання суміші нормальних розподілів є дескриптивним, іншими словами, у багатьох випадках може дійсно надійно описувати дані.

Перед тим як використовувати цей клас моделей треба ввести новий термін «схована змінна», вона може приймати наступні значення:

$$z = \{1, \textit{тоді } i - \textit{тий нормальний розподіл генерує } x_i; 0, \textit{в іншому випадку}\}$$

Для кожного спостереження є своя схована змінна, саме вона описує до якого саме нормального розподілу належить дане спостереження.

Щоб знайти центри розподілу можна скористатися EM-алгоритмом (Expectation-maximization алгоритм). Він є ітераційним.

Коли використовується GMM модель для задачі класифікації часто користуються тим, що спостереження різних класів можуть належати до не однакових розподілів. Після отримання параметрів за допомогою EM-алгоритму, можна для всіх спостережень провести класифікацію, класифікуючи кожен об’єкт по найбільшій імовірності.

1.2.5 Метод з використанням наївного баєсевого класифікатора

Наївний баєсів класифікатор — ймовірнісний класифікатор, він використовує в собі теорему Баєса для того, щоб визначити ймовірність приналежності елемента вибірки до класу з припущення(наївного) незалежності змінних. [5]

Основою узагальненого байесовського класифікатора, що працює являється правило: класифікатор обраховує ймовірність P(k|x) кожного класу k , до якого може належати об'єкт який досліджують, а потім створює залежність цього об'єкту з найбільш імовірним класом $\hat{k}$

$$\hat{k} = \arg\max \ln P(k \mid x_1,...x_m)$$

Ця імовірність називається апостеріорною та обчислюється за такою формулою:

$$P(k \mid x_1,...,x_m) = P(k)\,p(x_1,...,x_m \mid k) / p(k)$$

, де $P(k)$ - це ймовірність того, що об'єкт може відноситися до k -того класу, а $p(k)$ і $p(x_1,...,x_m \mid k)$ - це безумовна, а також умовна щільності розподілу вектору ознак, які є багатовимірними, компоненти вектору є залежними.

Таким чином, байесовський класифікатор припускає, що багатовимірна спільна щільність розподілу ознак відома для всіх класів.

Аналітично подати багатовимірну щільність імовірностей можна лише за умови нормального розподілу. Також багатовимірна нормальна щільність розподілу надає зручну модель ще і для випадку, коли значення у вектора ознак x для даного класу k завжди мають свої значення. Узагальнений байєсів класифікатор можна віднести до оптимальних класифікаторів. Доказом цьому є твердження, що якщо є однозначна відповідь, якщо така можлива, класифікатор її визначить.

Також плюсом його використання є те що на практиці в більшості випадків нестроге виконання припущення що змінні незалежні може призводити лише незначні втрати точності, тобто — переваги цього класифікатора, а саме простота і масштабованість, висока швидкість роботи та помірні вимоги до пам'яті часто переважують всі його недоліки.

1.2.6 Метод з використанням дерев рішень

Цей метод належить до самих популярних методів вирішення задачі класифікації. В його основі, як зрозуміло по назві закладений процес рекурсивної розбивки множини вхідних даних або об'єктів на підмножини, які є асоційованими із заданими класами.

Конструювання дерева рішень функціонально.

Виконується побудова дерева двома етапами. Спочатку створюється дерево, потім відбувається його скорочення.

Критерії зупинки, а також скорочення вибираються завчасно сам процес навчання відбувається під час першого етапу. На другому етапі відсікаються зайві гілки дерева.

Підбір критерію розщеплення.

Дерево створюється згори до низу. Під час такого процесу алгоритм повинен знайти потрібний параметр розщеплення(також називають критерій розбивки), за допомогою нього система розбивається на множини, асоційовані з поточним вузлом. Кожен із цих вузлів відповідає за окрему характеристику.

Основним правилом вибору критерію є те, щоб після розбиття даних скоротити кількість об’єктів, які належать не до свого класу, тобто випадкових невідповідностей.

Найбільш відомим у світі data science коефіцієнтом є індекс Джіні, а також міра ентропії. Міра ентропії тут — це значення, яке вказує наскільки класи, які знаходяться в одному вузлі відрізняються. Після того як створяться вузли з меншою

різноманітністю станів результату. Кажучи інакше, ентропія повинна спадати, а от кількість інформації, яка знаходиться в вузлах поступово збільшується.

Признак найкращого атрибуту є те, що він максимально покращує збільшення корисної інформації в вузлах дерева.

Якщо дерево стає більшим це не являється признаком його більшої ефективності в класифікації. Коли дерево збільшується в ньому збільшується кількість випадків, які були описані. Тож, в загальному, менша кількість спостережень буде віднесена до всіх класів. Дерева такого типу називаються "гіллястими". Їхній розмір не виправданий, створюється дуже маленькі підмножини з вхідної, та їх виходить занадто багато. Узагальнювати в таких деревах набагато тяжче. Тут можна привести аналогію із перенавчанням, коли створені моделі можуть працювати некоректно з новими даними, при цьому працюючи стабільно на тренувальному сеті.

Коли ми будуємо дерево потрібно приймати сукупність деяких мір, щоб результуюче дерево не вийшло занадто великим. Для цієї задачі використовують різні підходи, які називаються методами створення оптимальних дерев.

Щоб зрозуміти ці підходи потрібно дати визначення оптимальності дерев. Оптимальним можна назвати дерево, яке являється достатньо складним для вирішення задачі класифікації, проте не занадто складним.

Таке дерево повинне використовувати лише інформацію, яка покращує роботу класифікації і повністю ігнорувати інші параметри, які на покращення роботи ніяк не впливають.

Для знаходження цього використовують 2 підходи перший із них створює дерево розміру, який було задано користувачем в алгоритмі.

Інший підхід користується певними методами щоб визначити розмір оптимального дерева. Одним із таких методів є метод відсікання гілок, який по черзі рекурсивно відсікає гілки поки не знайде оптимальний момент щоб зупинитися, цей момент вираховується алгоритмічно.

Не всі методи конструювання дерев працюють однаково. Є, які завжди послідовно використовують тільки два етапи, що згадувались раніше(подудова і скорочення), є і такі, які чергують їх, скорочують дерево після того як воно побудоване, а потім знову будують вже на скороченому дереві. Останній варіант допомагає не допустити нарощуванню дерева, коли ж перший нарощує це дерево, а потім скорочує його.

Обрахування моменту зупинки побудови дерева

Тож, щоб визначити правило зупинки будування дерева, треба сформулювати умови виконання побудови цього дерева. Це правило має визначати, чи є даний вузол вже кінцевим , чи тільки внутрішнім.

Зупинкою можна назвати момент коли подальші розгалуження повинні перестати створюватись. Є декілька існуючих на даний момент правил обрахунку зупинки.

Першим методом є "рання зупинка". Тут розглядається лише математична доцільність розбивання робочого вузла. Значною перевагою такого методу є значне прискорення часу виконання навчання. З іншої сторони недоліком може стати те, що таке дерево стане гіллястим і втратить значну частину потрібної інформації в вузлах та фінальний результат матиме критично низьку точність передбачення.

Також можна обмежувати глибину дерева. В цьому алгоритмі побудова дерева повинна закінчитись при глибині зарані вказаній користувачем в алгоритмі відповідного методу.

Також існують правила зупинки коли завдається мінімальна кількість прикладів, які будуть знаходитись в останніх вузлах вихідного дерева. Коли кожен вузол дерева буде містити не менш ніж задане число об’єктів будування дерева припинеться.

Методи та підходи до скорочення дерева та відсікання в ньому гілок.

Як згадувалось раніше, основним способом запобігання гіллятості дерева і збільшення кількості невагомих вузлів є відсікання віток.

Класифікатор, який є навченим деревом рішень можна оцінити точністю розпізнавання помилок. Відсікання самих гілок дерева логічно проводити там і тільки там, де після відсікання помилка дерева не збільшиться і точність залишиться на тому ж рівні.

Даний процес йде від кінця дерева тобто його кінцевих вузлів до його вершини. Скорочення використовується на практиці частіше ніж правила зупинки. Після того як всі невагомі вітки відсіклися дерево називають "усіченим".

1.2.7 Метод з використанням штучних нейронних мереж

Штучними нейронними мережами називаються обчислювальні системи, які базуються на тих же ідеях на яких працюють біологічні нейроні мережі, які працюють всередині мозку.

Особливістю цього методу є те, що вони збільшують свою продуктивність на тренувальних даних і не потребують спеціальної програмної реалізації для конкретних випадків.

Для прикладу, щоб розпізнати зображення чи відеоряд нейронні мережі можуть просто навчитися класифікувати зображення, наприклад на вхід будуть подаватись картинки котиків і собачок із метаданими, які пояснюють нарисований котик чи собачка. Результати навченої моделі потім можна використовувати для класифікації зображень на яких мережа не тренувалась. Навчання відбувається без жодних знань про те як виглядають собачки чи котики, які вони мають вуха, скільки лап, яке хутро, який хвіст, які очі, і т.д. Замість всіх цих характеристик нейронна мережа створює свій набір характеристик базуючись на залежності цифрових даних, відносного розміщення пікселів для різних картинок підкорюються залежностям, які нейронні мережі знаходять за допомогою своїх алгоритмів.

Штучні нейронні мережі базуються на створенні з'єднаних вузлів, їх називають штучними нейронами (назва знову була взята із біології). Нейрони можуть передавати сигнали один одному за допомогою з’єднань між ними. Всі

нейрони можуть проводити унікальну для себе обробку вхідного сигналу та передавати його далі іншим нейронам.

Сигнали частіше всього в штучних нейронних мережах представляють із себе дійсне ціле значення. Вихід із всіх нейронів частіше всього обраховується, як нелінійна функція суми по входам конкретного нейрона.

Всі з'єднання мій нейронами мережі мають свою вагу, яка вона завжди змінюється під час навчання. Вага зв'язку впливає на силу сигналу, що передається. Часто штучні нейрони можуть мати поріг, якщо значення на вході нижче нього сигнал ігнорується.

Частіше всього мережі складаються з шарів, які складаються з певної кількості нейронів. Шари відрізняються способом перетворення сигналів на своїх входах. Вхідні сигнали проходять через кожен шар нейронні мережі змінюючись в її вузлах, деколи вони можуть проходити декілька раз через ті самі шари доки дійдуть до останнього.

Основна ідея штучних нейронних мереж це розв'язання задач класифікації способом, яким би це розв'язувала жива істота. Із часом розвиток нейронних мереж відійшов від ідей біологічних і зосередився більше на розвитку особливих здібностей ШНМ. Нейронні мережі використовуються в різних областях таких як розпізнавання людської мови, комп'ютерний зір, гра в відеоігри, машинний переклад тексту, медичні діагностування, машинний переклад тексту, соціально-мережеве фільтрування, тощо.

Найпопулярнішим та найбільш використовуваним видом нейронних мереж є багатошарові нейронні мережі[6]. Суть цієї архітектури в з'єднанні нейронів в декілька шарів із одним вектором вхідного сигналу. Вектор на вході подається на вхідний шар ШНМ. Виходами всієї нейромережі являються вихідні сигнали з останнього шару їх називають також ефектори.

Архітектура цієї мережі базується на:

1. Можливості мережі збільшуються в залежності із збільшенням кількості

вузлів та шарів в мережі.

2. Також збільшується потужність про додаванні зворотних зв’язків, проте тоді виникає динамічна нестійкість.

3. Якщо ускладнити алгоритми, за якими працює мережа, наприклад додати кілька типів синапсів, потужність мережі також зростає.

Пошук необхідних і достатніх властивостей мереж для розв’язування різного роду задач є окремим напрямом нейрокомп’ютерної науки.

Зазвичай проблема створення нейронної мережі залежить від конкретної задачі, яка розв’язується, тому з’явилась проблема утворення загальних рекомендацій. Найчастіше найбільш вдалий для використання варіант виходить із емпіричного підбору або використовуючи в основі інші вдалі рішення, які вже показали свою високу продуктивність на інших схожих задачах та дослідженнях.
Типовими архітектурами нейронних мереж є:

Мережі прямого розповсюдження:

- мережа зустрічного розповсюдження;

- мережа зворотного розповсюдження;

- карта Кохонена;
- перцептрони;

- згорткові архітектури;

Рекурентні мережі:

- двоспрямована асоціативна пам’ять;
- мережа Хопфілда;
- LSTM мережа;
- мережа адаптивної резонансної теорії;

Робота нейромережі, функціонування задач, які вона повинна розв'язувати, основується в основному на величинах зв’язків між її нейронами, а тому, при певній

архітектурі нейронної мережі, яка побудована для конкретної задачі , треба знайти всі значення коефіцієнтів,при яких вона працює оптимально.

Навчанням моделі називається етап на якому підбираються коефіцієнти ваги зв’язків. Чим краще буде навчана модель, тим кращі результати вона буде показувати при реальному її використанні для передбачень класів об’єктів.

Важливим тут фактором крім правильного і якісного вибору ваг є також і час навчання нейромережі. Зазвичай ці параметри залежать один від одного обернено пропорційно і на практиці приходиться нехтувати одним із них на користь іншого.

Може існувати навчання нейронних мереж як з вчителем так і без нього. Коли навчання проходить із вчителем нейронній мережі надаються вхідний клас, а також потрібний вихідний, якщо мережа передбачила його не правильно ваги коректуються деяким внутрішнім алгоритмом. Другий випадок виставляє ваги самостійно базуючись тільки на вхідних даних і даних від них похідних.

Алгоритми машинного навчання бувають різні та в основному їх поділяють на детерміновані і стохастичні. Перший алгоритм має чіткий алгоритм та правила виставлення ваг, тоді як у другому ваги виставляються випадковим чином.

Також нейронні мережі можна поділити на бінарні і аналогові. В першому випадку мережі оперують двома значеннями і на вхід будь-якого нейрона мережі може поступити тільки два значення: логічний нуль та логічна одиниця. Перцептрон також належить до цього виду нейронних мереж так як виходи його нейронів формують значення 0 або 1. В аналоговій версії виходи здатні приймати безперервні значення, що мало мати б місце, якщо замінити функцію активації перцептрону на сигмоїду.

Також нейронні мережі поділяються на синхронні та асинхронні. В першому випадку обрахунки в нейронах здійснюються по черзі, в другому можливе обрахування нейронів відразу шарами, або групами нейронів великої кількості одночасно.

Багатошарова архітектура є найпопулярнішою та найбільш застосовуваною в сучасному світі, її структура складається з декількох шарів нейронів кожний з яких з’єднаний із кожним нейроном наступного шару. Такі мережі називають повносв’язними(англ. Dense) [6].

В найбільш простому випадку, якщо мережа одношарова, алгоритм досить простий, в його основі лежить ідея, що відразу відомі правильні вихідні значення і зв’язки модифікуються так, щоб в результаті зменшувалася помилка на виході. На такому ж підході базується алгоритм навчання одношарового перцептрону.

Коли мережа має більше ніж один шар, немає інформації про оптимальні значення проміжних шарів. Перцептрон у якого більше 1 шару неможливо буде навчити такому алгоритму. Варіантом вирішення такої проблеми може бути створення різних наборів сигналів, які відповідають вхідним, повністю для кожного шару нейронної мережі. На практиці цей варіант не можливий так як потребує багато обрахунків та ресурсів.

Ця проблема також вирішується динамічним підлаштуванням коефіцієнтів ваг синапсів, під час цього процесу слабкі зв’язки збільшуються або зменшуються на невелике значення, зберігаються тільки ті значення, які сприяли зменшенню помилки на виході.

Метод на перший погляд здається досить простим, але знову на практиці потребує величезної кількості обчислень, так як шарів в подібній архітектурі зазвичай достатньо багато.

Використовують же на практиці ще один - третій варіант. Його суть полягає в тому, що сигнали помилки із виходів нейронної мережі поширюються до входів. Такий алгоритм навчання нейронної мережі називається процедурою зворотного поширення.

1.3 Метрики для оцінювання якості класифікації

Оцінюючи різні моделі машинного навчання виникає потреба визначення того, як же саме потрібно оцінювати результат виконання алгоритму. На сьогоднішній день є багато різноманітних метрик, сам вибір правильної метрики є вкрай важливим. Зазвичай, якщо одна із моделей виконує поставлену задачу набагато краще за іншу, то це означає, що ця модель переважатиме іншу по більшості метрик. Дивлячись інакше, якраз переважання по метрикам і може стати критерієм, що повідомляє нам модель працює набагато краще.

Крім цього існують випадки, коли за одною метрикою модель показує кращі результати проте гірші за іншою метрикою. Саме по цій причині при побудові алгоритму важливо визначитись із тим, яка саме метрика буде відігравати визначальну головну роль.

Для прикладу, якщо проводити наукову роботу буде доцільно використовувати однакову метрику для різних алгоритмів, щоб правильно зрівняти отримані результати. При комерційній розробці алгоритму машинного навчання, потрібно провести глибоке навчання предметної області і можливо навіть створити свою метрику, яка буде правильно характеризувати результати класифікації. Якщо вибрати метрику неправильно це може коштувати підприємству величезних збитків в вигляді витрачених людино-годин, часу на навчання алгоритму та на дослідження.

Найчастіше на практиці використовують такі метрики:

1.3.1 Точність

У найпростішому випадку метрикою може бути частка документів за якими класифікатор прийняв правильне рішення.

$$\text{Accuracy} = \frac{P}{N}$$

, де P – число документів при яких класифікатор прийняв правильне рішення, а N - розмір тренувальної вибірки.

Варто враховувати одну особливість даної метрики.

Метрика надає повністю всім документам рівну вагу, це може бути не правильно у випадку якщо розподіл документів в початковій вибірці дуже зміщений в сторону якихось класів. Тоді в такому випадку у класифікатора є набагато більше даних про ці класи і зрозуміло, що в рамках даних класів він буде робити більш правильні рішення.

В реальному світі це почитає призводити до того, що вироблений алгоритм має загальну точність, скажімо, 85%, але з якимись із класів класифікатор працює дуже погано, не вгадуючи навіть третини із вього тренувального сету. Також можна навести приклад, коли єдва класи. Один з них становить 90% від всієї тренувальної вибірки, а інший відповідно тільки 10%. Виходить, що такий класифікатор може мати точність 0.90, правильно прогнозуючи лише перший класс. Це не те чого має хотіти розробник алгоритму машинного навчання. В тому випадку, якщо приклад не абстрактрий і реальних класів більше два, та розташовані вони вкрай нерівномірно, по мій метриці оцінити якість алгоритму майже неможливо.

Рішенням такої проблеми може бути правильно збалансовані вхідні, а також тестові дані. Мінусом такого рішення є затрати по часу на підготовку даних, та і зачасту необхідної кількості даних просто може не бути.

Для збалансовування класів можна використовувати зменшення кількості прикладів класу, яких більше в датасеті, або можна згенерувати нові приклади для класів, кількість яких мала.

1.3.2 Повнота та прецизійність.

Повнота та прецизійність це метрики, які використовують зазвичай при оцінці в більшості випадків алгоритмів для отримання корисної інформації. Деколи ці метрики використовують просто самі по собі, деколи в якості основи для інших похідних від них метрик, наприклад F-міра. Суть цих метрик надзвичайно проста.

Системна точність в середині класу - це частина документів, які правда належать до даного класу щодо об’єктів, яких система віднесла до конкретного класу.

Повнотою називається частка знайдених класифікатором об’єктів, що належать класу щодо всіх об’єктів цього класу в даній тестовій вибірці.

Значення повноти і точності можна злегкістю розрахувати по таблиці контингентності вона складається окремо для кожного класу, або ж через матрицю заплутувань, вона буде розглянута пізніше.

Приклад такої таблиці наведений в таблиці 1.

В даній таблиці знаходиться основна інформація, вона необхідна щоб оцінити прецензійность та повноту, це інформація, яка вказує скільки разів система зробила вірне, а також скільки разів система прийняла не правильне рішення по об’єктах даного класу.

Складається таблиця із чотирьох комірок, всі комірки містять один варіант класифікування можливий для одного спостереження.

Конкретніше:

TP (англ. true positive) - істинно-позитивний вибір, алгоритм класифікатору оприділив, що спостереження належить до певного класу і його рішення співпала з експертною;

TN (англ. true negative) - істинно-негативний вибір, алгоритм класифікатору оприділив, що спостереження не належить до класу, і його рішення так само співпало з експертною;

FP (англ. false positive) - хибно-позитивний вибір, алгоритм класифікатору оприділив, що спостереження належить до класу, але нажаль відповідь опинилась не вірна;

Таблиця 1 - Приклад таблиці контингентності.

Категорія I		Експертна оцінка	
Оцінка системи		Позитивна оцінка	Негативна оцінка
	Позитивна оцінка	TP	FP
	Негативна оцінка	FN	TN

FN (англ. false negative) - хибно-негативний вибір, алгоритм класифікатору оприділив, що спостереження не належить до класу, але воно належить;

Тоді, прецизійність і повнота визначаються наступним чином:

Прецизійність - $\text{Presition} = \frac{\text{TP}}{TP + FP}$

Повнота - $\text{Recall} = \frac{\text{TP}}{TP + FN}$

1.3.2 Матриця заплутувань

Матрицею неточностей називають матрицю розміру n×n , де n це кількість класів кластерів. Стовпці такої матриці мають значення експертних рішень, а рядки це рішення класифікатора користувача. Коли класифікується об’єкт із тестового сету збільшується число, яке стоїть рівно на перетині рядка класу передбаченого класифікатором і стовпця до якого реально відноситься об’єкт(рис. 1).

Маючи цю матрицю повноту та точність для всіх класів можна досить просто розрахувати. Точність можна порахувати, як відношення діагонального елемента матриці на суму всіх елементів рядка класу. Повнота рахується як відношення діагонального елемента матриці до суми всіх елементів стовпця класу

Класифікаторну прецизійність можна розрахувати середнім арифметичним точності по всіх його класах. Так само рахується повнота. З технічної точки зору даний підхід називається macro-averaging.

	0.91	0.96	0.94	0.75	1.00	0.83	0.85	0.97	1.00	0.86	1.00	0.79	1.00	0.75	1.00	1.00	0.96	0.90	0.81	0.89	0.94	0.98	0.86	0.89	0.94	0.92	0.96
0.80		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0.95	1	94	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
1.00	2	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.29	3	0	0	6	0	0	3	2	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1	3	0	2	0
1.00	4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.50	5	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2	0	1	1
0.92	6	1	0	0	0	0	152	0	0	1	0	0	0	0	0	0	0	1	4	2	3	0	0	0	0	2	0
0.97	7	1	0	1	0	0	0	256	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	2	0
0.33	8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0.97	9	0	0	0	0	0	0	0	0	69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0.82	10	0	0	0	0	0	2	0	0	0	18	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0.87	11	0	0	0	0	0	0	0	0	0	0	34	0	4	0	0	0	0	0	0	0	0	0	1	0	0	0
1.00	12	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.57	13	0	0	0	0	0	0	0	0	0	0	9	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0
0.63	14	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	3	0	0	0	0	0	0	0	0	0
0.50	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	1	0	0	0	0	0	0
0.77	16	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	47	0	1	3	4	0	0	2	0	1	0
0.87	17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	69	1	2	5	0	0	0	0	0	0
0.97	18	0	0	0	0	1	4	0	0	1	0	0	0	0	0	0	0	0	197	1	0	0	0	0	0	0	0
0.78	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	35	183	13	0	0	2	0	1	0
0.97	20	0	0	0	0	0	10	3	0	1	0	0	0	0	0	0	0	0	0	4	702	0	0	0	0	6	0
0.93	21	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	2	0	0	0
0.29	22	0	0	1	0	0	2	0	0	6	0	0	0	0	0	0	0	0	1	1	1	0	6	2	0	1	0
0.91	23	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	3	6	0	0	115	0	0	0
1.00	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
0.93	25	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	4	5	0	0	0	1	196	0
0.98	26	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	78

Рисунок 1 – Приклад матриці заплутувань

2 ЗАСОБИ РОЗРОБКИ

Важливим чинником, під час розробки програмного продукту, є вибір засобів програмної реалізації та технологій. Середовищем розробки інформаційної системи було обрано Jupyter Notebook.

Для розробки алгоритмів використовувалась об’єктно-орієнтована мова програмування Python.

2.1 Середовище розробки Jupyter Notebook

Jupyter Notebook— комерційне інтегроване середовище розробки для різних мов програмування (Python, Java, Scala, PHP та ін.) дуже зручне для створення аналітичних звітів так як дозволяє зберігати в собі разом код, зображення, графіки, коментарі і формули. Magic команди можна гнучко використовувати разом із мовою програмування наприклад команда %run дозволяє виконувати програмні файли з розширенням .py прямо із Notebook, також в ньому можна викликати будь яку shell команду, що зручно для керування віртуальним середовищем. Середовище має зручну систему роботи з часом, налагодження коду через %debug. Маркдаун комірки дозволяють підмальовувати формули LateX за допомогою MathJax. Також в одному ноутбуці може використовуватись одразу декілька мов програмування використовуючи відповідні макроси, для цього потребується лише просте налаштування середовища.

2.2 Мова програмування Python

Python – високорівнева мова програмування загального призначення, орієнтований на підвищення продуктивності розробника і читабельності коду.

Синтаксис ядра Python мінімалістичний. В той час коли стандартна бібліотека включає в себе тисячі корисних функцій. Python підтримує структурне, об’єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване програмування. По зрівнянню з строго типізованими мовами Python в багато разів збільшує продуктивність роботи розробника [3].Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети. Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках, включаючи пропрієтарні. Є реалізація інтерпретатора для JVM з можливістю компіляції, CLR, LLVM, інші незалежні реалізації. Проект PyPy використовує JIT-компіляцію, яка значно збільшує швидкість виконання Python-програм.

2.3 Бібліотеки Keras і Tensorflow

Keras - відкрита нейромережева бібліотека, написана на мові Python. Вона являє собою надбудову над фреймворками Deeplearning4j, TensorFlow і Theano. Націлена на оперативну роботу з мережами глибинного навчання, при цьому спроектована так, щоб бути компактною, модульною та з можливістю розширюватися. Вона була створена як частина дослідницьких зусиль проекту ONEIROS. Планувалося що Google буде підтримувати Keras в основний бібліотеці TensorFlow, однак Шолль виділив Keras в окрему надбудову, так як згідно з концепцією Keras є скоріше інтерфейсом, ніж наскрізний системою машинного навчання. Keras надає високорівневий, більш інтуїтивний набір абстракцій, який

робить простим формування нейронних мереж, незалежно від використовуваної в якості обчислювальної бекенд бібліотеки наукових обчислень.

TensorFlow - відкрита програмна бібліотека для машинного навчання, розроблена компанією Google для вирішення завдань побудови і тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття. Застосовується як для досліджень, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою реалізований для Python, також існують реалізації для C Sharp, C ++, Haskell, Java, Go і Swift. Є продовженням закритого проекту DistBelief. Спочатку TensorFlow була розроблена командою Google Brain для внутрішнього використання в Google, в 2015 році система була переведена в вільний доступ з відкритою ліцензією Apache 2.0.

2.4. Модуль PyQt

PyQt - це набір Python v2 та v3 для рамкової програми Qt Company Qt Company і працює на всіх платформах, підтримуваних Qt, включаючи Windows, macOS, Linux, iOS та Android. PyQt5 підтримує Qt v5. PyQt4 підтримує Qt v4 і Qt v5. Прив’язки реалізовані у вигляді набору модулів Python та містять понад 1000 класів.

PyQt4 і Qt v4 більше не підтримуються, і нові релізи більше не розробляються. PyQt5 та Qt v5 настійно рекомендуються для всіх нових розробок.

PyQt об'єднує міжплатформенну прикладну програму Qt C ++ і інтерпритовану мову Python.

Qt - це більше, ніж набір інструментів GUI. Він включає абстракції мережевих сокетів, потоків, Unicode, регулярних виразів, баз даних SQL, SVG, OpenGL, XML, повністю функціональний веб-браузер, систему довідки, мультимедійні рамки, а також багату колекцію віджетів GUI.

Класи Qt використовують механізм сигналу / слотів для зв'язку між об'єктами, який є безпечним для типу, але вільно поєднаним, що спрощує створення повторно використовуваних програмних компонентів.

Qt також включає Qt Designer, графічного дизайнера інтерфейсу користувача. PyQt здатний генерувати код Python за допомогою Qt Designer. Також можна додати нові елементи управління графічним інтерфейсом, написані на Python до Qt Designer.

Python - це проста, але потужна об'єктно-орієнтована мова. Його простота дозволяє легко навчатися, але її потужність означає, що можна створювати великі і складні програми. Її інтерпретована природа означає, що програмісти Python дуже продуктивні, оскільки не існує циклу розробки редагування / компілювання / посилання / запуску.

Значна потужність Python походить від його всебічного набору модулів розширення, що забезпечують широкий спектр функцій, включаючи HTTP-сервери, XML-парсери, доступ до бази даних, засоби стиснення даних і, звичайно, графічний інтерфейс користувача. Модулі розширення зазвичай реалізуються або в Python, C або C ++. Використовуючи такі інструменти, як SIP, порівняно прямо вперед можна створити модуль розширення, який інкапсулює існуючу бібліотеку C або C ++. Використовуваний таким чином, Python може потім стати клеєм для створення нових програм із створених бібліотек.

PyQt поєднує всі переваги Qt та Python. Програміст володіє всією силою Qt, але здатний використовувати його простотою Python.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі міститься обгрунтування архітектурного рішення при розробці системи класифікації гідроакустичних сигналів. Також розділ обгрунтовує використання Pandas Dataframe.

3.1 Кластеризація

Вхідний датасет містить більше 7 тисяч гідроакустичних сигналів, довжина яких 10 секунд, частота дискретизації 2048Гц, кількість каналів 4. Кластеризація вхідного датасету була проведена за допомогою виділення MFFC(мел-кепстральні коефіцієнти) з кожного сигнала, для задачі кластеризації достатньо брати декілька коефіцієнтів в нашому випадку було взято 10 коефіцієнтів.

Збережені характеристики зберігаються в Pandas Dataframe з 11 колонками, 10 коефіцієнти та 1 з назвою відповідного файлу. Далі отриманий dataframe передається в функцію fit_predict перед цим нормалізуючи значення коефіцієнтів для більш коректних результатів, fit_predict розбиває вхідний масив на задану кількість класів, цю кількість знаходимо, як оптимальне значення за допомогою «правила ліктя» це коефіцієнт при якому середня квадратична похибка перестає знижуватись експоненціально, в нашому випадку це 6.

Знайдений розподіл класів зберігаємо у .csv файлі з метаданими, де міститься назва файлу її лейбла класу та MFFC коефіцієнти.

Функціональна декомпозиція системи кластерізації наведена на рисунку 3.1.

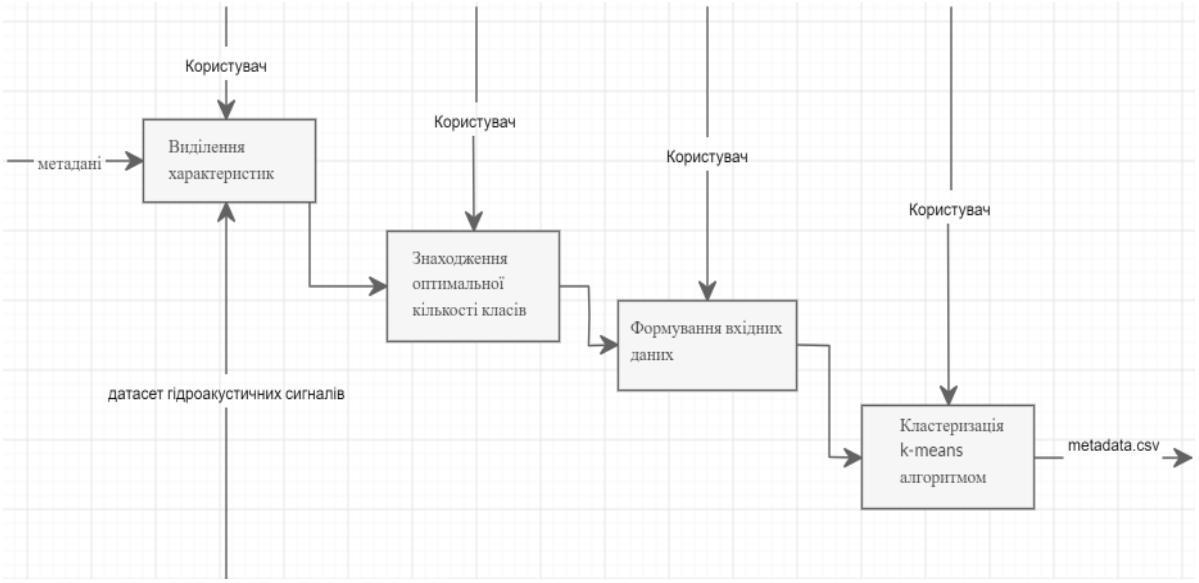

Рисунок 3.1 - Фукціональна декомпозиція системи кластеризації

3.2 Класифікація MLP

Класифікація датасету виконується на базі отриманих після кластеризації метаданих з розподілом сигналів по класах. Спочатку виділяємо характеристики звуку(MFFC) з аудіофайлів та зберігаємо ці дані, а також дані про назву класу в числові масиви, які є зрозумілими для комп’ютера для того щоб навчити моделі.

Для початку спробуємо створюється модель в архітектурі MLP використається Keras та Tensorflow. Модель будується шар за шаром.

Почнемо з простої архітектури яка складається з трьох шарів: вхідного, вихідного і одного прихованого. Всі шари будуть мати стандартний тип 'dense'

Перший шар буде отримувати вхідну форму. Так як файли містять 40 MFCC ми матимемо форму 1x40 це означає ми почнемо з вхідної форми з 40 входами.

Перші два шари будуть мати 256 вузлів. Будемо використовувати функцію ReLU(Rectified Linear Activation) як функцію активації для перших двох шарів, ця

функція довела свою працездатність в багатьох нейромережах успішно працюючих на сьогоднішній день.

Dropout встановлюється на 50% для перших двох шарів. Це дозволить випадково видаляти вузли з кожного прогону циклу для того щоб покращити результати передбачення нейромережі і зменшить шанс на "overfit".

На виході будемо мати 10 вузлів які будуть збігатися з нашими класами звуків. Функція активації для вихідного шару 'Softmax', вона сумує вихідні значення до 1 так, що результат можна оцінювати як ймовірність, модель буде робити свої припущення базуючись на цій ймовірності.

Для компіляції використовуються наступні параметри:

- Loss function - categorical_crossentropy. Найчастіше використовується для класифікації. Менше значення цього параметру означає, що модель рахує краще.
- Metrics - accuracy метрика яка дозволить нам переглядати точність передбачень моделі.
- Optimizer - adam хороший бібліотечний оптимізатор.

Нейромережа навчається протягом 21 хвилини.

Функціональна декомпозиція підсистеми класифікації наведена на рисунку 3.2.

3.3 Pandas dataframe як засіб зберігання даних

Pandas Dataframe — це складний проіндексований масив значень. Загрузити значення в нього можна за допомогою різних джерел даних таких як SQL, текстові дані, html, або Excel. Фізично Dataframe представляє із себе таблицю над наборами даних в ній можна виконувати будь якого роду дії, об’єднання, додавання стовпців, записів, фільтрація та інше. Ця структура повністю замінює реляційні бази даних для проектів в області досліджень Data science. В програмному продукті

використовується декілька датафреймів для збереження метаданих, таблиць із характеристиками кластерізації, темпових даних.

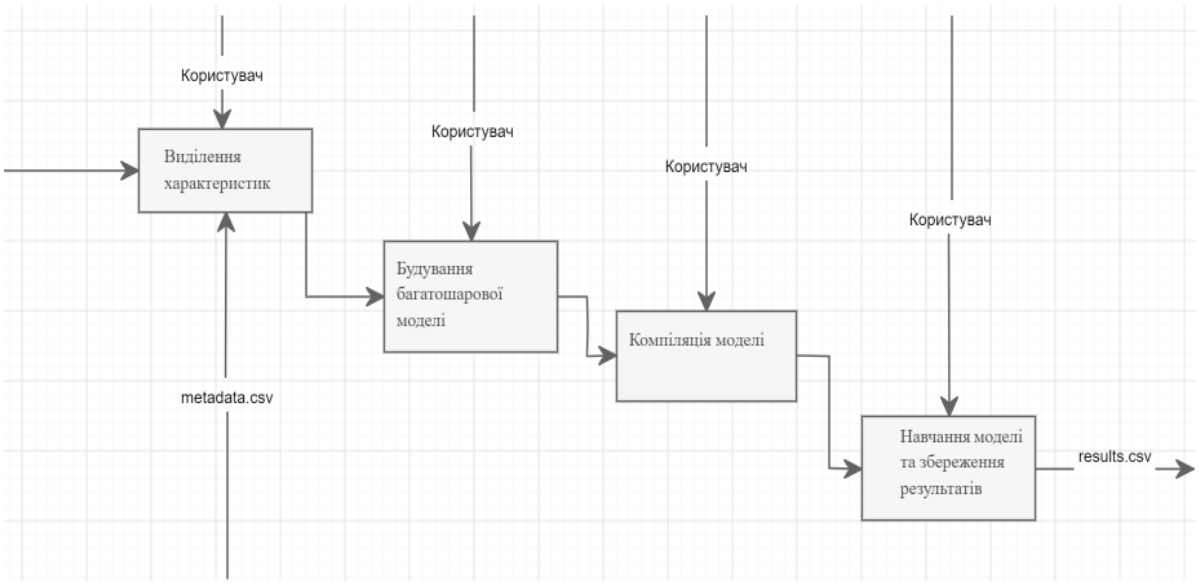

Рисунок 3.2 – Функціональна декомпозиція підсистеми класифікації

В данній роботі Pandas Dataframe використовується для збереження проміжних даних в вигляді складних таблиць, а також для збереження результату класифікації та кластеризації та їх імпортування в excel таблиці.

На рисунку 3.3 приведено вигляд проміжних метаданих, які є результатом роботи модулю кластеризації даних.

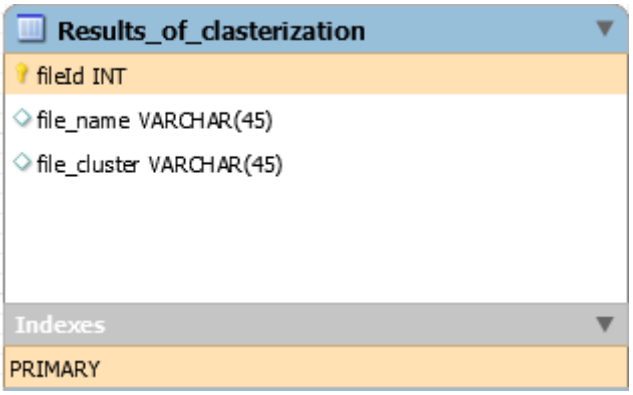

Рисунок 3.3 – Таблиця метаданих

На рисунку 3.4 зображено таблиця в якій зберігаються результати роботи класифікатора для даного датасету.

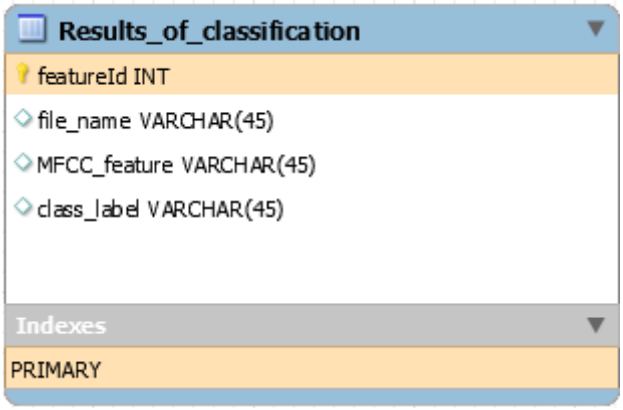

Рисунок 3.4 – Таблиця результатів

Перевагою бібліотеки Pandas перед іншими засобами збереження даних є зручність її використання, інтуїтивно зрозумілий синтаксис, чудова інтегрованість із середою розробки для відображення результатів прямо в Jupyter Notebook та можливість експортування даних в різних форматах включаючи зручні таблиці excel(.csv)

4 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В цьому розділі приведені системні вимоги для роботи з додатком та сценарії роботи користувача з ним.

4.1 Системні вимоги

Для забезпечення коректної та безвідмовної роботи системи класифікації гідроакустичних сигналів персональний комп'ютер повинен мати процесор не гірше, ніж Intel ® Pentium ® / Celeron ® / Xeon™ або з тактовою частотою не менше 1,8 GHz або AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ для користувачів процесорів від фірми AMD. Також комп'ютеру користувача повинно бути доступно не менше 2 Gb оперативної пам'яті та графічне ядро не гріше, ніж Intel ® HD Graphics 2000, що еквівалентно графічним картам з об'ємом пам'яті не менше, ніж 128 Mb також для даного програмного забезпечення потрібно не менше 1.5 Гб вільного місця на жорскому диску.

Також в користувача повинен бути встановлений Jupyter Notebook, Python 3.6.4 64 bit(так як бібліотека tensorflow не сумісна із версіями вище та з 32 бітними версіями) та наступні пакети: tensorflow version 1.12.0, Keras 2.2.4, pandas 1.0.3, numpy 1.18.4, scipy 1.4.1, sklearn 0.0, librosa 0.7.2.

4.2 Взаємодія користувача з програмним продуктом

Для успішної кластеризації датасету, користувач повинен скористатися Jupyter Notebook #1 DataClasterization1.ipynb, потрібно запустити всі комірки доки не дійде до комірки з заголовком «Знаходження оптимальної кількості класів»

Запустивши цю комірку після того, як користувач побачив графік виведеним в інтерфейсі Notebook він повинен з графіку відношення квадратичної похибки sse до к знайти оптимальне к, знаходиться воно «методом ліктя» знайшовши к при якому похибка перестає зменшуватись експоненціально, для випадку наведеному на скріншоті це число рівне 6(Рисунок 4.1).

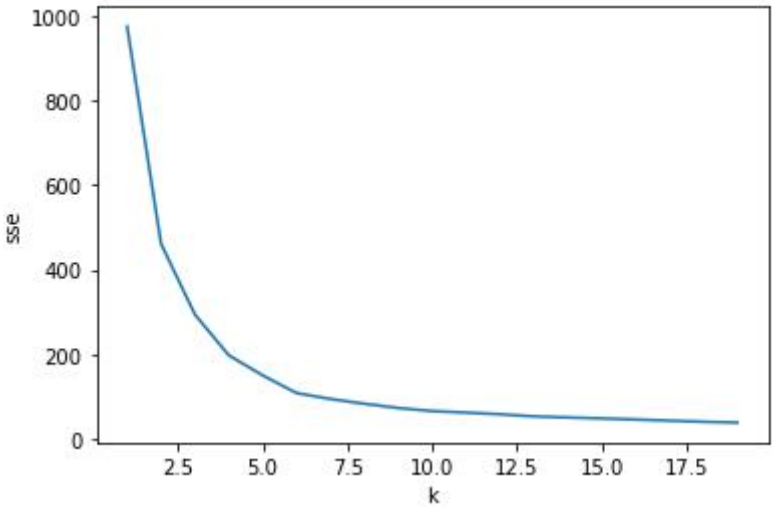

Рисунок 4.1 – Графік залежності кв. похибки від k

Далі користувач даного Notebook повинен виконувати скріпти з можливостю перегляду отриманих результатів, наступний момент втручання можливий в пункті «Кластеризація по n_clasters» де користувачеві потрібно ввести бажану кількість класів для розбиття датасету на рисунку 4.2 показаний приклад роботи з 6 кластерами.

Результат розбиття класів на кластери можна переглянути в інтеративному режимі за допомогою одної із відповідних комірок в графічному вигляді(Рисунок 4.3), або у вигляді таблиці.

Кластеризація по n_clusters класах

```
In [ ]: #Lets add more information to dataframe
import sklearn
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder


km = KMeans(n_clusters=6)
y_pred = km.fit_predict(DFeature[['feature1', 'feature2', 'feature3', 'feature4', 'feature5', 'feature6', 'feature7', 'feature8'
y_pred
```

Рисунок 4.2 – Кластеризація датасету на k класів

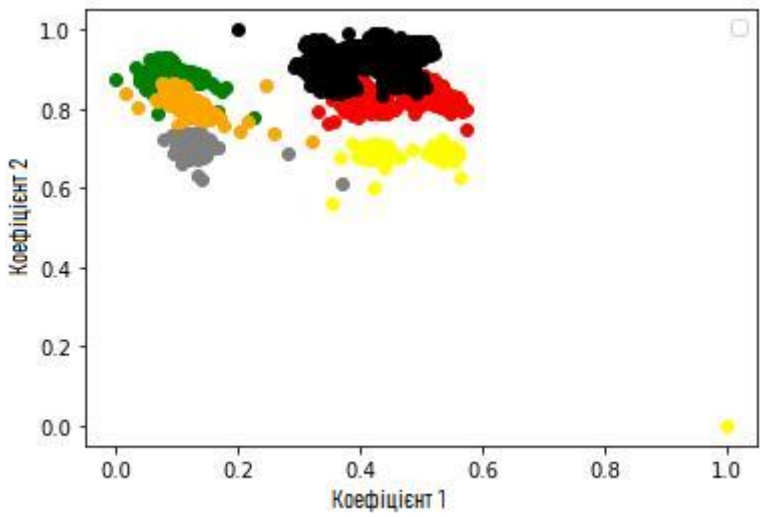

Рисунок 4.3 – Розподіл аудіо сигналів по класам в координатах перших двох характеристик кластеризації

Після зберігання метаданих даних про приналежність сигналів до конкретних класів можна приступати до процесу будування моделі класифікації.

Першим для цього процесу потрібно запустити другий Notebook під назвою DataPreprocessingAndDataSplitting.ipynb, кожна комірка зі скріптом чітко прокоментована і не потребує втручань користувача, на цьому етапі підготовлюються характеристичні дані для навчання моделі. Наступним потрібно відкрити останній Notebook ModelTrainingAndEvaluation.ipynb за допомогою якого проходить навчання мультилеєр нейромережі. Процес навчання мережі(Рисунок 4.4) складається зі 100 епох.

```
Train on 5896 samples, validate on 1475 samples
Epoch 1/100
5896/5896 [==============================] - 3s 531us/step - loss: 12.3725 - acc: 0.2232 - val_loss: 11.2772 - val_acc: 0.300
3

Epoch 00001: val_loss improved from inf to 11.27720, saving model to saved_models/weights.best.basic_mlp.hdf5
Epoch 2/100
5896/5896 [==============================] - 2s 272us/step - loss: 11.2254 - acc: 0.3034 - val_loss: 11.2772 - val_acc: 0.300
3

Epoch 00002: val_loss did not improve from 11.27720
Epoch 3/100
5896/5896 [==============================] - 2s 313us/step - loss: 11.2165 - acc: 0.3041 - val_loss: 11.2772 - val_acc: 0.300
3

Epoch 00003: val_loss did not improve from 11.27720
Epoch 4/100
5896/5896 [==============================] - 2s 255us/step - loss: 11.2192 - acc: 0.3039 - val_loss: 11.2772 - val_acc: 0.300
3
```

Рисунок 4.4 – Процес навчання мережі

Після того як користувач «навчив» модель (Рисунок 4.5) він може

```
# Evaluating the model on the training and testing set
score = model.evaluate(x_train, y_train, verbose=0)
print("Training Accuracy: ", score[1])

score = model.evaluate(x_test, y_test, verbose=0)
print("Testing Accuracy: ", score[1])
```

```
Training Accuracy: 0.88
Testing Accuracy:  0.81
```

Рисунок 4.5 – Результати точності оцінювання

використовувати комірку із методом print_prediction() для передбачення належності сигналу до одного із зарані отриманих класів(Рисунок 4.6)

```
In [15]: print_prediction('..\..\sigwav\signal_124.wav')
         The predicted class is: class_3
```

Рисунок 4.6 –Передбачення вихідного класу

Для наглядності виведення результатів класифікації було створено додаток для виведення класу сигналу по введеній назві файлу гідроакустичного сигналу. Вигляд інтерфейсу користувача наведений на рисунку 4.7.

Щоб вивести результат класифікації потрібно ввести шлях до сигналу на натиснути на кнопку «Класифікувати», результат на рисунку 4.8 де сигнал було віднесено до класу «class_2».

Результат виводу цього додатку повністю залежить від кроків описаних раніше і не дасть ніякого результату без проходження етапу кластеризації та класифікації в середовищі Jupyter Notebook.

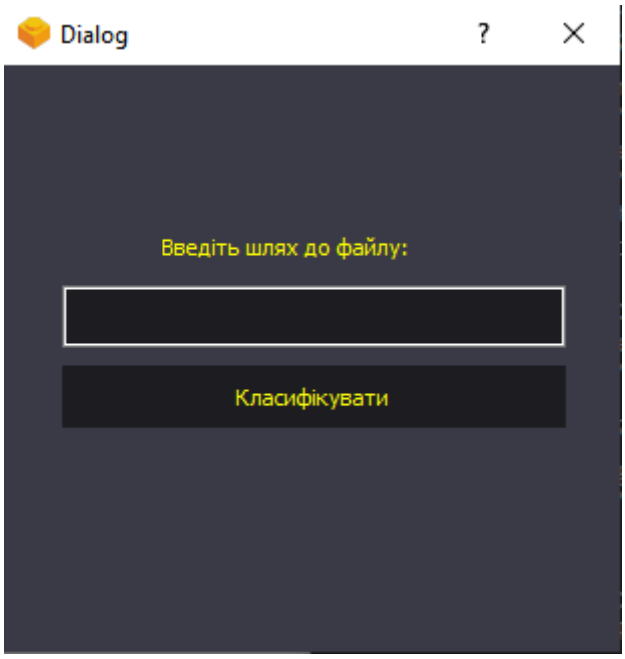

Рисунок 4.7 – Вигляд GUI результатів класифікування

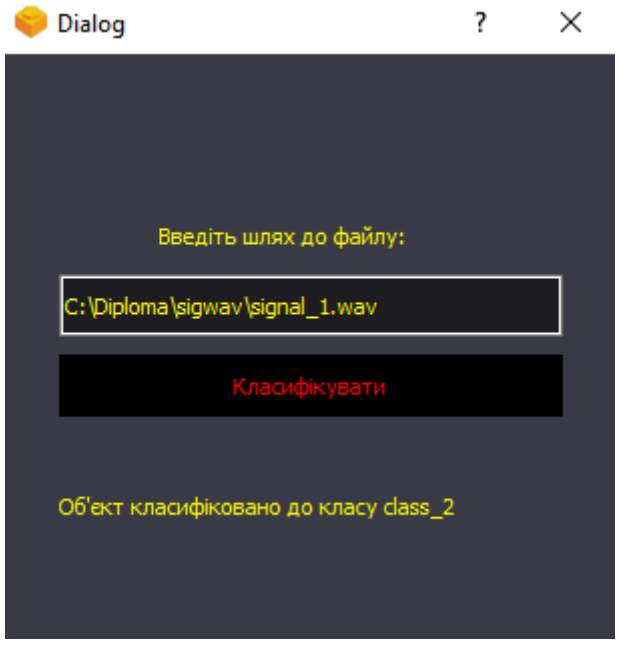

Рисунок 4.8 – Результат виконання класифікації сигналу

ВИСНОВКИ

В ході роботи було проведено огляд та зроблено аналіз засобів, що були використані для створення даного програмного забезпечення (середовища розробки Jupyter Notebook та засобів створення: Python, Tensorflow, Keras).

Програмний продукт було написано на мові програмування Python з використанням середовища Jupyter Notebook.

В ході виконання даної роботи було розроблено систему класифікації гідроакустичних сигналів(модуль попередньої обробки сигналів, модуль кластеризації та модуль класифікації).

Програму можна використовувати для класифікації гідроакустичних сигналів.

Для роботи з даним програмним забезпеченням необхідний комп’ютер середньої потужності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сидоров К. В. Анализ признаков эмоционально окрашенной речи / К. В. Сидоров, Н. Н. Филатова. // Вестник ТвГТУ. – 2012. – №20.

2. Николенко С. И. Конспект лекцій з предмету Automated Speech Recognition [Електронний ресурс] / С. И. Николенко. – 2008. – Режим доступу до ресурсу: https://logic.pdmi.ras.ru/~sergey/teaching/asr/notes-06- features.pdf

3. Trevor H. The elements of statistical learning. Second edition / H. Trevor, R. Tibshirani., 2008.

4. Шеремет О. Метод опорних векторів / О. Шеремет, В. Садовой. // Мат. Мод. № 1. – 2013. – №28.

5. Domingos P. On the optimality of the simple Bayesian classifier under zero- one loss / P. Domingos, P. Pazzani. // Machine Learning. – 1997. – №29.

6. Біла Н. І. Інформаційні системи та технології в управлінні. Теоретичні відомості і завдання до лабораторних робіт / Н. І. Біла.

7. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville., 2016.

8. Zhang W. Shift-invariant pattern recognition neural network and its optical architecture / Wei Zhang. // Proceedings of annual conference of the Japan Society of Applied Physics. – 1988.

9. Zhang W. Parallel distributed processing model with local space-invariant interconnections and its optical architecture / Wei Zhang. // Applied Optics. – 1990. – №29. – С. 4790–4797.

10. Subject independent facial expression recognition with robust face detection using a convolutional neural network / M.Matusugu, M. Katsuhiko, Y. Mitari, Y. Kaneda. // Neural Networks. – 2003. – №16. – С. 555–559.

11. Николенко С. И. Глубокое обучение. Погружение в мир нейронных сетей. / С. И. Николенко, А. А. Кадурин, Е. О. Архангельская., 2018.

12. Glorot X. Understanding the difficulty of training deep feedforward neural networks / X. Glorot, Y. Bengio. // Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. – 2010.

13. Krizhevsky A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. Hinton. // Advances in Neural Information Processing Systems. – 2012. – №1.

14. Dropout: A Simple Way to Prevent Neural Networks from overfitting / [S. Nitish, G. Hinton, A. Krizhevsky та ін.]. // Journal of Machine Learning Research. – 2014. – №15. – С. 1929–1958.

15. Salamon J. A Dataset and Taxonomy for Urban Sound Research / J. Salamon,

C. Jacoby, J. Bello. // 22nd ACM International Conference on Multimedia, Orlando USA. – 2014.

16. ImageNet: A Large-Scale Hierarchical Image Database / [J. Deng,W. Dong, R. Socher, L.-J. Li та ін.]. // CVPR09 – 2009

17. Gwardys G. Deep image features in music information retrieval / G. Gwardys, D. Grzywczak. // International Journal of Electronics and Telecommunications. – 2014. – №60. – С. 321–3